International Journal of Clinical Monitoring and Computing 7: 137–146, 1990.

Development of decision support systems

T. Allan Pryor
Med. Biophysics & Computing, University of Utah, LDS Hosp./325 Eight Ave, Salt Lake City, UT 84143, USA

Accepted 6 February 1990

Abstract

Use of hospital information systems (HIS) are no longer limited to administrative functions. The addition to these systems of decision support capability is now a necessity. Development of the decision support modules requires a different software architecture than that employed by most HIS systems today. This paper describes the generic uses of decision support throughout the many hospital applications. Several levels of decision support are outlined with examples to illustrate the many areas where decision support is useful. At LDS Hospital in Salt Lake City, Utah we have developed an HIS using a new software architecture which supports the creation of decision support applications. This system uses a frame structure to represent knowledge. Examples of the frames and their syntax is presented. Using the frame tools which are provided, an application developer can easily develop and test decision support modules which interact directly with the clinical user and the patient database.

Use of decision support systems within a hospital are becoming almost a necessity. As Hospital Information Systems (HIS) usage has grown in the hospital the requirements of such a system to assist in the decision processes of the medical personnel has simultaneously grown. The fact that data is now available on the computer has stimulated those with access to that data to demand increased software to utilize the data in their decision support requirements. These requirements have come not only from the medical staff, but include the nursing, administrative and ancillary staff as well. In response to those requests system developers have used both the power of the PC for quickly developing decision support programs and attempted to upgrade existing HIS's to incorporate decision support software. Because of the need to integrate all decision support programs with an integrated hospital database, we at LDS Hospital have chosen to develop an HIS which is knowledge driven for the express purpose of being a decision support system which also performs all of the traditional HIS functions.

The HELP system [1–4] at LDS Hospital differs greatly from other hospital information (HIS) and decision support systems currently available. It differs first from traditional HIS systems in that it incorporates decision support mechanisms in every HIS application. It also differs from the expert systems reported in the literature in the breadth of decision domains and methods which it handles. Expert systems such as DXPLAIN [5], MYCIN [6], Internist [7] are limited to a single decision model and application. These programs use a particular model for implementing expert logic and attack a single decision problem such as diagnosis. The HELP system on the other hand is a system where multiple decision support models and applications can coexist. Therefore, there is no single decision support model which exclusively defines the HELP decision support methodology. Within HELP the use of decision support is not limited to a

single application. As explained below HELP is designed to support the entire application needs of a hospital; i.e. administrative, clinical and financial.

Decision support uses

In design of the HELP system we first investigated where decision support was used within the hospital. This investigation was necessary to ensure that the design we chose to implement the system would provide the necessary tools for easy creation of decision support software in every required application. We discovered that different from traditional program design a decision support HIS must allow the expert to manage the knowledge of the system. This investigation also lead us to define six major generic uses of decision support in the hospital which are:

1. Alerting
2. Interpretation
3. Assisting
4. Critiquing
5. Diagnosing
6. Managing

Understanding these generic decision needs leads to the development of a flexible HIS which allows creation of decision supported applications which serve the entire user population of an HIS.

Alerting decision support is defined as automatic notification of the appropriate personnel of a time critical of action oriented decision. The most common clinical example where alerting decision support is incorporated is medication ordering systems [8]. In these systems alert criteria is evaluated at the time the order is entered into the system and an alert generated if the criteria of the alert is met. The alert generally is in the form of a message on the terminal to the ordering person before subsequent orders may be entered. This alert may indicate a drug-drug interaction, an allergy contraindication or some important clinical laboratory interaction/ contraindication. For example, an alert may be generated when ordering a potassium sparing drug in a patient whose laboratory values already indicate the patient has a low potassium value. In this instance the alert may suggest the need to order potassium concurrently with the potassium sparing agent. Beyond the use of alerting decision support in pharmacy systems the same alerting techniques can be extended to include notification of the nurse on any abnormal trends sensed from nurse charted data, management alerts which could notify nursing of failure to complete some required nursing task, or alerts for notification of hospital administration of a DRG cost over run on a patient, etc. One of the features of most alerting systems is their ability to monitor data in the background and create alerts as the appropriate criteria is met. Bradshaw [3] describes and evaluates such a background laboratory alerting system. Alerting systems have easily been the most used mode of decision support in the hospital. Virtually every hospital information system today probably has some simple or sophisticated alerts in one or more of its applications.

Interpretation refers to assimilation of data resulting in a conceptual understanding of the data. The mode of decision support has also been widely used for many years in the hospital. The most common application of interpretation found in hospital computers systems today is the computerized interpretation of the ECG. Early in the development of the HELP system ECG interpretive system was developed [9]. Several commercial companies including Hewlett Packard and Marquette Electronics offer ECG interpretive systems. With these systems the ECG is not only recorded by the instrument, but is analyzed to determine both the morphological and rhythm status of the patient. A report is then generated which provides the actual waveforms, measured values from the waveforms and the clinical interpretation of the waveforms. Another example of an interpretive decision support use is the interpretation of Blood Gas data [10]. As with the ECG systems the interpretive blood gas programs report to the physician not only the measured values from the blood, but an interpretive report of the meaning of those values. Many of today's instruments used in hospitals are equipped with microprocessors which accomplish some level of interpretation of the data they are processing.

Assisting decision support is decision support

used to speed or simplify some human interaction with the computer system. Assisting decision support usually incorporates predictive knowledge about the task to be performed. Clinically assisting decision support is commonly used in systems to enter physician orders. For those systems which are intended for use by the physician to enter his/her own orders assisting decision support is a necessity if the program is to be sufficiently efficient for use by the busy physician. In this example the assisting decision support would use predictive knowledge to suggest the appropriate order parameters for a given patient and order. This could be in the form of a fixed standing order list, calculated parameters of the order such as patient specific dose rates for a given drug, fixed or patient specific protocols for care of the patient, etc. In each of these examples the assisting decision makes use of logic available to the application to reduce the number of steps normally taken in the ordering process. This reduction usually translates to a system which is then sufficiently rapid to make it viable for the use by the time conscience physician. Prokosch [2] has reported on the use of an assisting decision support system to aid in physician ordering of medications. Likewise predictive knowledge can be easily built into the nurse charting applications, again making those computer tasks simpler and more efficient. While many systems provide for some sort of fixed default/predictive knowledge/tables in the ordering process, the use of more complex assisting logic is still primarily a research effort.

Critiquing decision support is defined as the computerized analysis of human suggested decisions to verify the appropriateness of the decision. This form of decision support has been widely suggested in reviewing physician entered clinical orders or patient management plans. This form of decision support is distinguished by the formulation of the order or plan. In the critiquing mode it is assumed that no decision support was provided to suggest the order/plan, but merely to use the knowledge base to evaluate the order/plan and report to the user the result of that computerized evaluation. One example of critiquing decision support by Miller [11] is the management of ventilator settings on a respirator. In this example the physician enters into the critiquing program a suggested set of respirator settings. The program using its critiquing logic evaluates the suggested settings against the patient's computer stored findings and the rules of the critiquing logic. The results of the critique are then presented to the physician for review and action. The results could be agreement with the initial suggested plan or present reasons why such a plan may be detrimental to the patient. Under the later conditions the program would suggest more appropriate settings with some justification of its reasoning. Except in the simplest form of critiquing medication orders similar to the alerting mode, critiquing decision support remains a research issue.

Diagnostic decision support is the use of decision support to generate diagnostic suggestions about the patient/system. This form of decision support has probably received more attention in the literature than any other form of decision support. With these programs the user generally enters some initial signs, symptoms and laboratory values from which the program begins hypothesizing diagnoses. In order to conclude on a diagnosis the programs generally begin a dialogue with the user to gather the most pertinent data which it believes is necessary to conclude an appropriate diagnosis. While the logic of the different diagnostic programs vary, one of the most distinguishing features between them is the number of diseases they consider. Programs such as DXPLAIN [5] may consider several thousand diseases, while another program may be limited to only pulmonary diseases. In general the utility and accuracy of the programs is commensurate with the number of diseases it considers. As the number of diseases rise, the speed of the program generally increases and the ease of use of the program declines. Because of the complexity of these programs, the authors have explored their use in teaching environments where the student may interact extensively with the program to gain experience in the diagnostic process. Warner [12] has described such a system called ILLIAD. Diagnostic decision support can be used to create applications not only for traditional patient diagnosis, but where models exist for understanding any other aspect of the hospital environment, the decision

support can be applied to help the user understand the state of any system in the hospital.

Finally, management decision support is the automatic generation of action oriented decisions designed to improve the system state. Management decision support differs from critiquing decision support in that with management, decisions suggest treatment plans whereas in critiquing support the computer is reacting to treatment plans entered by the physician. With management decisions the roles of the computer and physician have changed. The physician takes the role of critiquing the computer rather than the computer critiquing the physician. Clinically management decisions have taken the form of simple or complex protocols encoded in the computer. Using the logic of these protocols the computer can monitor the state of the patient and make suggestions regarding subsequent treatment. With the exception of some small closed loop medication administration systems, the suggestions of the computer are always reviewed by the user before implementation of the computer logic. The use of management decision support in nursing is also being investigated by several researchers. In particular the automatic creation of nursing care plans is being developed at several institutions. In this application the computer working off the patient's diagnosis would automatically suggest to the nursing staff a care plan which could be accepted or modified by the nurse. The level of sophistication of management decision systems are dependent on the amount of patient specific logic contained in the programs. While those with fixed protocols are easy to implement, without tailoring the protocols to the individual patient, they may be too simple to serve as valuable assistance. Sittig [13] reports on an extensive protocol management system used in the ICU's to assist in the management of patients with Adult Respiratory Distress Syndrome.

System design to implement decision support uses

As noted, our research into the use of decision support in the hospital led us to believe that 1) there is need for widespread use of decision support in the hospital and 2) the system must permit the expert to manage the decision support logic in his domain. To research these goals HELP, therefore, was designed as a knowledge driven HIS which executes expert defined and maintained knowledge modules (frames). This required, therefore, developing for HELP a set of knowledge tools which the expert could use for definition and maintenance of their decision logic. These knowledge tools would provide access to the knowledge base in a form that is easily understood and controlled by the expert and implemented in an efficient lower level machine executable form. Two primary knowledge tools were developed. The first was a knowledge editor to create and maintain the logic and the second was a knowledge compiler to generate the machine executable programs. With this design we hoped to create a system which is both maintainable by the expert and efficient for the user. Figure 1 portrays this process. As outlined in the figure, the expert interfaces with the knowledge base through general or domain specific knowledge editors for creation/maintenance of the source knowledge frames. The knowledge frames are then compiled into a general purpose query language for execution on the computer. The compiled knowledge frames reside in the object knowledge base and can be randomly executed when needed in the decision support applications of the HIS. Because all of the knowledge frames exist independently in a common knowledge file, a single knowledge frame may be shared/executed by several HIS applications.

To assist the expert in creation/maintenance of his expert logic we had to design a decision support language (frame language) which would be used to write the knowledge frames. The knowledge editors were created to also enhance the ability of the domain expert to manage his/her expert logic. In designing our frame language the first goal was to make the language simple enough to be understood by non computer experts and yet complete enough to perform the logic required by the decision support applications. The language had to support four major feature/functions which were deemed necessary in programming a complete decision support application. These features are 1) the ability to interface knowledge variables to the HIS database,

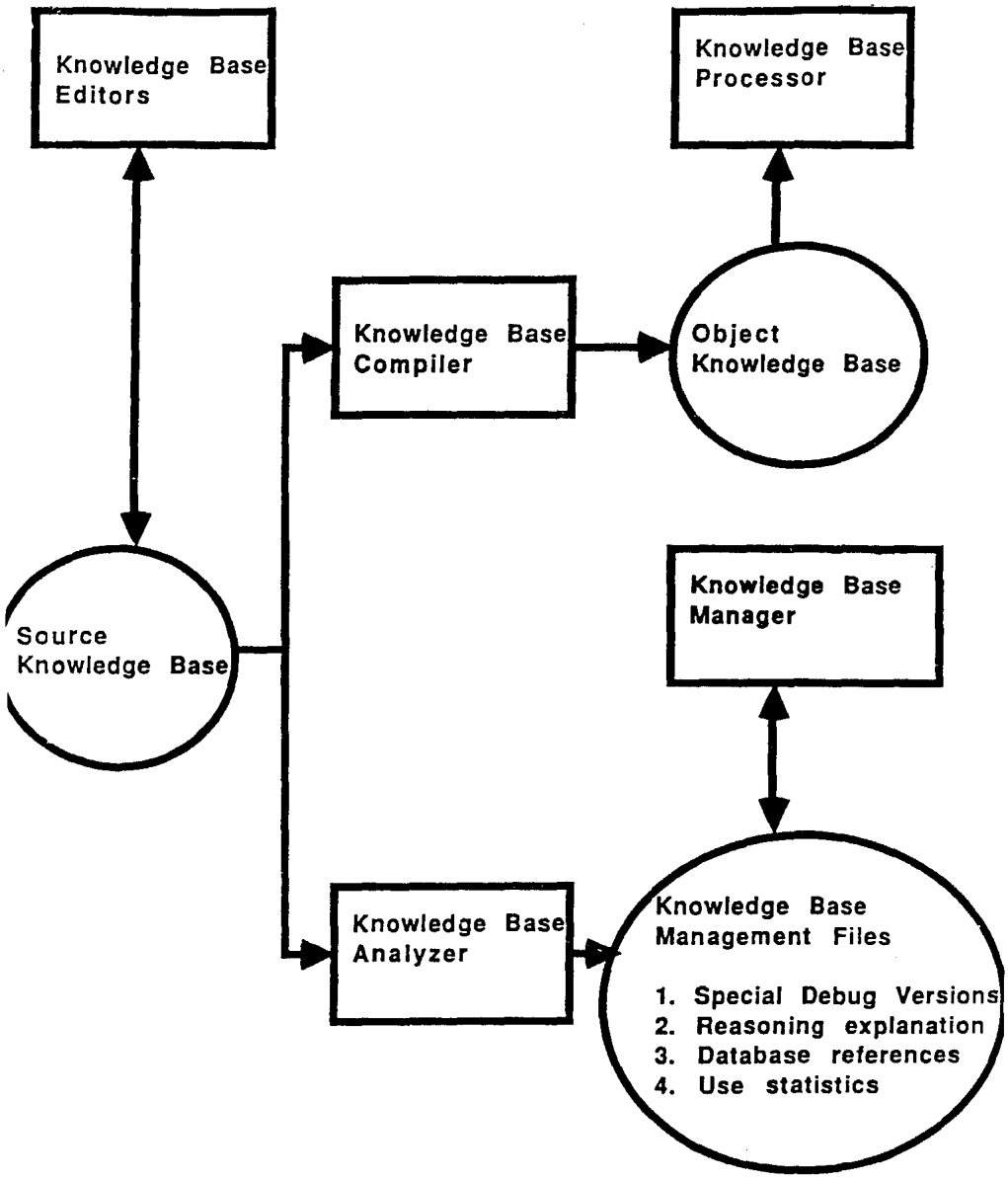

Fig. 1. Block diagram of the knowledge frame generation process. Source frames are first created using special knowledge editors. These frames are then compiled into object code using a special purpose frame compiler to a general query language on HELP. The object knowledge base consists of modular randomly accessible frames of knowledge ready for execution by the HELP knowledge editor. The knowledge base manager is used to analyze the contents of the frame for maintenance, use and explanation purposes.

2) the ability to define data acquisition methodologies, 3) the ability to define data/decision reporting requirements, and 4) the ability to write the actual decision logic. Given these requirements we developed a frame structure where the frame slots are segmented into the following components: 1) Frame management knowledge slots, 2) Attribute knowledge slots and 3) Declarative/Procedural knowledge slots.

Figure 2 illustrates the slots defined for frame management knowledge. The slots of the knowledge segment allow us to track the source, validity, utility and type of the knowledge represented in the frame. Figure 3 is an example of a frame where the slots have been instantiated with the particular values illustrated in the figure. As seen in Fig. 3 not all of the available slots need be instantiated in a frame. While most of the slots are self explanatory, the frame type, gold standard, validation level, and utility slots require some explanation. The frame type slot defines the intended primary use of the frame. Several frame types have been declared.

FRAME MANAGEMENT KNOWLEDGE

Author:

Title:

Frame Type:

Create/Update Data:

References:

Gold Standard:

Validation Level:

Utility:

Comments:

Fig. 2. The slots in the knowledge frame used for management purposes.

They include management frames, diagnostic frames, data acquisition frames, and report frames. The gold standard slot contains knowledge about any gold standard which could be used to measure the accuracy of the frame. The gold standard slot could refer to another frame, a particular test, a discharge diagnosis, etc. That is any criteria which could be used by a knowledge management system to validate the knowledge of this frame. The validation slot contains knowledge concerning the level of validation through which the frame has been tested. For example, is this logic only the best guess of the knowledge engineer who wrote the frame or does the logic come from some literature reference or has the logic been tested in a controlled study, etc. The utility slot is used to record the importance

Title: Aminophyllin bolus in asthma (continuous drip in separate frame) (7.126.3)

Author: Peter Haug

Type: Management

Message: "Suggest asthmatic treatment begin with an aminophyllin IV bolus of <dosage> mg."

Utility: 5/9

Reference: Washington Manual of Medical Therapeutics

Fig. 3. The knowledge management slots from a typical decision frame.

Declare Variables: asthma **which is** REACTIVE AIRWAY DISEASE (ASTHMA),

aminophyllin_allergy **as an expression containing** theophyllin
which is THEOPHYLLINES **and** theophyllin_containing_meds
which is THEOPHYLLINES IN COMBINATION WITH OTHER AGENTS
and drug_sensitivity **which is** DRUG SENSITIVITY
if drug_sensitivity **exists and** (theophyllin exists or
theophyllinp_containing_meds **exists**),

asthma_med_reaction **which is** HAVE YOU EVER HAD A REACTION TO A MEDICATION GIVEN FOR ASTHMA?

current_aminophyllin **as an expression containing** theophyllin
which is THEOPHYLLINES **and** theophyllin_containing_meds
which is THEOPHYLLINES IN COMBINATION WITH OTHER AGENTS
and current_med **which is** CURRENT and home_med **which is**
HOME MEDICATION
if (theophyllin **exists or** theophyllin_containing_meds **exists**) **and**
(current_med **exists or** home_med **exists**),

weight **which is** WEIGHT,

arterial_pO2 **which is** PO2,

Fig. 4. The attribute knowledge from a typical decision frame. The underlined text correspond to the actual database variables.

of the decision medically. That is, how clinically significant is the result of the logic of the frame.

The attribute knowledge of the frame contains knowledge about the relationship of the attributes/variables and the decision of the frame and knowledge about the database where the attributes/variables resides. We have also included as attribute knowledge, knowledge about the screen presentation of the attributes. This knowledge is in the form of window definitions for presentation/acquisition of the attributes/variables. Included in the attribute knowledge of the frame knowledge about attribute statistics such as sensitivity/specificity, ad hoc scores, allowed values, etc. are also included. For some decision frames (e.g. medication order prediction frames) attribute knowledge is sufficient to contain all the necessary knowledge of the frame. Figure 4 is the attribute knowledge segment of a frame intended to suggest treatment of Aminophyllin. In this example the underlined phrases represent individual elements which would be found in the patients database. The attributes/variables such as aminophyllin-allergy represent the attributes/variables which in this instance would be used in the declarative knowledge section of the frame.

The declarative/procedural knowledge of the frame contains the rules, equations and procedural flow of the frame. Control of terminal and/or database acquisition of the attributes/variables is contained in this section of the frame. In the example of Fig. 5 the logic statement contains a simple decision rule. The Ask slot contains control logic for the acquisition of variables which are needed for

Logic: **If** asthma **GE** 0.70 **and Not Exist** aminophyllin_allergy **and Not Exist**
current_aminophyllin **and Not** (asthma_med_reaction **EQ** yes) **Then** dosage =
5 *weight
Else Stop.

Ask: **Patient** (asthma_med_reaction) **Hierarchical.**

Evoke: **If** asthma **GE** 0.70.

Fig. 5. The declarative/procedural knowledge of a frame of suggesting use of aminophyllin.

subsequent execution of the frame. In this example the patient is to be asked about his/her reaction to asthma medications. The Evoke slot describes the logic necessary for automatic execution of this frame. In this example if the patient has at any time during his/her hospital stay noted the fact that he/she has a probability of asthma GE 0.70 stored in his/her computerized medical record, then this frame would automatically be processed by the system to determine the need for Aminophyllin treatment. In the case of background or data driven execution of the frame the resulting decision of the frame will be stored in the patient's computerized record.

We designed the syntax for the frame slots with the goal of insuring that the frame will be able to accomplish all the tasks necessary in implementation of a complete decision support application. Some of the features supported by the frame language syntax include 1) reference to the HELP database, 2) sophisticated screen presentation, and 3) support of different decision support models.

Database syntax in the HELP frame language allow the frame developer to easily retrieve complex variables from the patient database, reference other relational files supported by HELP, and record results of data entry and/or decisions in the patient database. Screen presentation syntax allows the use of multiple windows within a frame. This syntax is interfaced to a PC based window package which actually manages the windows and terminal data entry. Thus, the frame writer may easily define presentation attributes, allowed data entry ranges, validation procedures, and help screens within the context of the frame language. Finally, syntax is available in the frame language for easy implementation of simple logic rules. Bayesian decisions and other mathematical models. This ability of the frame system to support multiple decision models and screen presentations gives to the system a flexibility not supported in most decision support systems. While this adds complexity to the language, the knowledge specific editors provide a user friendly mode of interaction with the frames in those areas where a common model is desired for an entire application.

Development of the decision support applications

Development of the decision support applications now becomes a process of either writing new frames or utilizing existing frames in a new context. Most new applications, in fact, are a combination of the two methods. That is, existing frames are combined with new frames to constitute a new decision support application. Since the frame library is a common resource to all application developers, its scope and utility continues to grow as the new applications (frames) are added to the system. Of importance in this design concept is that not only are the source representation of the frames kept in the common library, but the object (executable) representation of the frames are kept in a common frame object library. Thus, maintenance of applications using shared frames is automatic with the modification of the shared frames. For example, a frame which monitors blood pressure trends and is used by both the application for automatic monitoring of blood pressure from bedside monitors and the nurse charting application for manual entry of blood pressure are automatically updated to new criteria when the blood pressure trends frame is changed to incorporate newer criteria.

To enhance the productivity of the application developers, several knowledge frame tools are available. The first is a general purpose frame editor. This editor is designed to easily permit the developer to write frames in the frame language. It is syntax dependent and requires the user to be conversant with the syntax of the frame language. Using this tool or a text editor, source knowledge frames can be created and transmitted to the frame compiler for compilation into the frame object representation. We have also written a series of special purpose frame compilers intended for use by domain experts who are not familiar with programming techniques. An example of such a special purpose editor is our general questionnaire editor. This editor permits the user to create data acquisition frames used with terminal data entry applications. The editor, through menus presented to the user, creates a frame with sophisticated windowing

```
TITLE:  NEURO MENU                                          [1 : 1 : 2];
MESSAGE:
AUTHOR:  I-0 0 0 0 0 0 0 0 - NANCY                                  -I;
FRAME TYPE:   DATA ACQUISITION;
MAINTENANCE
VARIABLE DECLARATIONS:
ORIENT        ;LOC          ;MOVE              ;GENBEHAV       ;DATABASE
    MOVES WHICH IS I-28 01 01 02 03 04-I;
    GENERAL WHICH IS  I-28 01 01 02 03 05-I;

WINDOW: NEURO                    (type (MC), minmax ( 0, 5),
    heading ("NEUROLOGICAL MENU", NORMAL),
    location ( 2, 0), control (NP),
BEGIN
FIELD:  ORIENT          (order (, 0),
    text ("Orientation", NORMAL),
    location ( 1, 2), input (N),
    default ( ),
    select (SN), control (NS),
    field help ( " ", " ") );
FIELD:  LOC
    text ("Loc", NORMAL),
    location ( 2, 2), input (N),
    default ( ),
    select (SN), control (NS),
    field help ( " ", " ") );
FIELD:  MOVE                     (order (,0) ,
    text ("Movement and strength", NORMAL),
    location ( 3, 2), input (N),
    default ( )
    select (SN), control (NS),
field help ( " ", " ") );
FIELD:  GENBEHAV                 (order (, 0),
    text ("General behavior", NORMAL),
    location ( 4, 2), input (N),
    default ( ),
    select (SN), control (NS),
    field help ( " ",   " ") ) ;
END;
 LOGIC:  ACQUIRE NEURO;
Pack;
 FOLLOWUP:  ALERT: =4;
    IF NOT $ EXIST ORIENT OR NOT $ EXIST LOC OR NOT $ EXIST MOVE OR
    NOT $ EXIST GENBEHAV OR NOT $ EXIST DATABASE THEN BEGIN
    STACK (ALERT, MOVES, GENERAL);
    EXIT FALSE;
    END;
    IF $ EXIST ORIENT THEN CALL  28 . 1 . 10;
    IF $ EXIST LOC THEN CALL 28 . 1 .  11;
```

Fig. 6. A frame data acquisition which includes windowing syntax in the frame language.

and permits definition of decision logic for evaluation of the entered data. Figure 6 is a typical frame created by this tool. The user does not need to know the syntax of the window/field statements, but is responsible for creation of Acquire, Diagnostic, and Follow-up logic. Acquire logic is that logic executed by the frame prior to presentation of the data acquisition screen. Diagnostic logic is that logic executed by the frame immediately following the completion of the data acquisition for purposes

of validating and/or error checking of the entered data. Follow-up logic is that logic executed by the frame to complete the decision function of the frame. Additional editors have been written for entry of medication requisition frames, general standing orders, etc.

In order to conveniently move to this knowledge frame based application model, we have interfaced the executable frame system to all existing languages running on HELP. Therefore, applications on HELP can evolve to completely frame based applications. For example, the pharmacy application may initially use the decision frames only for evaluation of medication contraindications by calling the appropriate frames following the entry of the medication order from the traditional non frame written pharmacy program. It can then evolve to calling predictive ordering frames from the same program by merely replacing the code in the pharmacy program for entry of the medication order with a call to a set of frames to assist in the medication order. This process allows to easily enhance our exixting applications without requiring a complete rewrite of the application as a frame based application.

All of the decision support applications on HELP are undergoing the transformation to this newer architecture of frame based decision support. As we continue the transformation we anticipate newer syntax/features to be added to the frame language. Effort is also underway to develop frame management programs which will assist us in the maintenance and understanding of the use of the knowledge frames as a model for a generalized decision support system.

References

1. Pryor TA, Gardner RM, Clayton PD, Warner HR. The HELP System. Journal of Medical Systems. Vol. 7 1983; 7: 87–101.
2. Prokosch HU, Pryor TA. Intelligent Data Acquisition in Order Entry Programs. Proceedings of the Twelfth Annual Symposium on Computer Applications in Medical Care. 1988: 454–458.
3. Bradshaw KE, Gardner RM, Pryor TA. Development of a Computerized Laboratory Alerting System. Computers and Biomedical Research. December 1989; 6: 575–587.
4. Pryor TA, Clayton PD, Haug PJ, Wigertz O. Design of a Knowledge Driven HIS. Proceedings of Eleventh Annual Symposium on Computer Applications in Medical Care. 1987; 60–63.
5. Barnett GO, Cimino JJ, Hupp JA, Hoffer EP. DXplain: An Evolving Diagnostic – Support System. Journal of American Medical Association. 1987; 258: 67–74.
6. Shortliffe EH. Computer Based Medical Consultion: MYCIN, American Elsevier New York, 1976.
7. Miller RA, Pople HE, Myers JD. Internist – 1, and experimental computer – based diagnostic consultant for internal medicine. New England Journal of Medicine. 1982; 307: 468–476.
8. Hulse RD, Clark SJ, Jackson JC, Warner HR, Gardner RM. Computerized Medication Monitoring System. American Journal of Hospital Pharmacy. 1976; 33: 1061–1064.
9. Pryor TA, Lindsay AE, England W. Computer Analysis of Seria Electrocardiograms. 1972; 6: 709–714.
10. Gardner RM, Cannon GH, Morris AH et al. Computerized Blood Gas Interpretation and Reporting System. Respiratory Care. 1985; 30: 695–700.
11. Miller PL. Medical Plan – Analysis by Computer: Critiquing the Pharmalogical Management of Essential Hypertension. Computers and Biomedical Research. 1984; 17: 25–40.
12. Warner HR, Haug PJ, Bouhaddou O, Lincoln M, Warner H Jr., Sorensen D, Williamson JW, Fan C. ILLIAD As An Expert Consultant to Teach Differential Diagnosis. Proceedings of Twelfth Annual Symposium on Computer Applications in Medical Care. 1988; 371–376.
13. Sittig D. Computerized Management of Patient Care in a Complex, Controlled Clinical Trial in the ICU. Proceedings of the Eleventh Annual Symposium on Computer Applications in Medical Care. 1987; 225–229.

Address for offprints:
T.A. Pryor,
Med. Biophysics & Computing,
University of Utah,
LDS Hosp./325 Eight Ave,
Salt Lake City, UT 84143, USA